# Man-Machine Cooperation for the On-Line Training of an Evolving Classifier

Manuel BOUILLON, Eric ANQUETIL
Université Européenne de Bretagne, France
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
IRISA, CNRS UMR 6074, Campus de Beaulieu, F-35042 Rennes
{manuel.bouillon, eric.anquetil}@irisa.fr

***Abstract*—Touch sensitive interfaces enable new interaction methods, like using gesture commands. To easily memorize more than a dozen of gesture commands, it is important to be able to customize them. The classifier used to recognize drawn symbols must hence be customizable, able to learn from very few data, and evolving, able to learn and improve during its use. This work studies the importance and the impact of using reject to supervise the on-line training of the evolving classifier. The objective is to obtain a gesture command system that cooperates as best as possible with the user: to learn from its mistakes without soliciting him too often. There is a trade-off between the number of user interactions, to supervise the on-line learning, and the number of classification errors, that require a correction from the user.**

## I. INTRODUCTION

With the increasing use of touch sensitive screens, human-computer interactions are evolving. New interaction methods have been designed to take advantage of the new potential of interaction that those interfaces offer. Among them, a new concept has recently appeared: to associate commands to gestures. Those gesture commands[1] [1][2] enable users to execute various actions simply by drawing symbols. Previous studies [3][4] have shown that enabling customization is essential to help user memorization of gestures. To use such gesture commands, a handwritten gesture recognition system is required. Moreover, if gestures are personalized, the classifier has to be flexible and able to learn with few data samples.

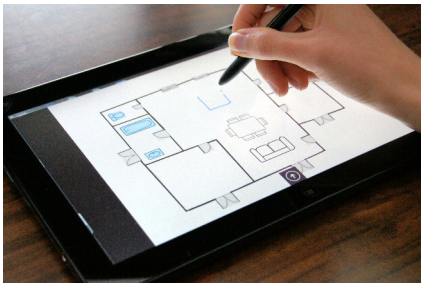

Fig. 1. Gesture command used to insert furniture in an architectural plan.

Gesture commands give rise to a cross-learning situation where the user has to learn and memorize his gestures and the classifier has to learn and recognize drawn gestures. Enabling customization of the gesture commands is essential for user memorization. On the other hand, enabling users to choose their own gestures may lead to commands with similar or strange gestures that are hard to recognize for the classifier. Moreover, we can't expect users to draw much more than a few gesture samples per class, so the recognition engine must be able to learn with very few data samples. Some template matching classifiers exist, like the $1 classifier [5] for instance, that don't require much training. However, such simple systems have limited performances, and don't evolve with the user writing style. For example, novice users usually draw gestures slowly and carefully, but as they become more and more expert, users draw their gestures more fluidly and rapidly. In that case, we want the classifier to adapt to the user, and not the other way round. More flexibility in a recognizer requires an on-line system, a system that learns on the run-time data flow.

Evolving classification systems have appeared in the last decade to meet the need for recognizers that work in changing environments. They use on-line learning algorithms to adapt to the data flow and cope with class adding (or removal) at run-time. This work uses such an evolving recognizer – namely *Evolve* [6][7] – which is a first order fuzzy inference system. It can start learning from few data and then learns incrementally in real time from the run-time data flow, to adapt its model and to improve its performances during its use.

The on-line learning algorithm is a supervised algorithm that requires labeled data. In the context of gesture command recognition, the only way of knowing the true label of a gesture is to interact with the user. However, soliciting the user after each command cancel the very interest of gesture commands! The method we use consist first, to take advantage of implicit validations of recognized labels by the user: if he continues his action without canceling or undoing the executed command, he implicitly validates the recognition. Secondly, we use the classifier self-evaluation capacity to solicit the user and obtain data true label when the confidence of the recognition is low.

The confidence measure of the classifier recognition can be of two kinds: an absolute measure for distance rejection or a relative measure for confusion reject. For the strategies presented in this article, we use an inner confidence measure evaluating the classifier confusion degree. This measure allows to reject data when they are between the classifier models of two classes, and that it will be very beneficial to learn from it. We are hence using confusion reject.

[1] See http://youtu.be/qOx4IY6uYf8 for a gesture commands demonstration.

Our objective is to handle as best as possible the cooperation between the user and the command gesture system to optimize the classifier training without soliciting the user too often. User interactions are tied to the rejection capacities of the classifier, to be able to learn from complex data that are hard to recognize. From a classification point of view, there is a trade-off between rejections and recognition errors. However, finding a compromise is particularly complex in such a on-line learning situation, where the classifier evolves continually. Rejecting a data sample that would have been badly recognized avoid a recognition error, but also supply an additional training data sample. In the article, we highlight the importance to find the best rejection strategy to optimize the man-machine cooperation. Indeed, a rejected gesture implies an user interaction, but provide a new labeled data for the on-line learning of the classifier. Therefore, it seems important to speed up the classifier training at the beginning of the command gesture system use, to reach a more interesting equilibrium afterwards. We developed a new strategy that rejects more at the beginning of the classifier training, and then less when the system has converged. This strategy increases the rejection rate at the beginning of the classifier training, to rapidly reduce the error rate, and then reduces the rejection rate to minimize user solicitations.

Another advantage of having a significant number of rejections at the beginning of system use is to avoid the "out-of-the-loop performance problem" [8]. This problem is the consequence of automation, here gesture recognition, without the operator having direct control. This situation can have harmful consequences like vigilance decrements or complacency. To avoid this problem, [9] propose to provide feedback to the operator on the automated task and the possibility to take control in case of failure. Rejection of complex data that are hard to recognize by the classifier hence allow to inform the user of system difficulties and to explicitly ask him to take control and correct the system (which he can also do in a spontaneous manner).

This paper is organized as follows. Section II presents the architecture of our evolving classifier, its incremental learning algorithm and the rejection capacity we introduced to develop our supervision strategy. We detail in Section III how our different strategies for the on-line training work, and their impact on user interactions. Then, we compare the different supervision strategies in a realistic experimentation in Section IV. Section V concludes and discusses future work.

## II. Evolving Fuzzy Inference System

This Section presents the evolving Fuzzy Inference System (FIS) on which this work is based [7]. We quickly describe the architecture of a first order FIS. Next, we present the incremental learning algorithm we use for the on-line training of our classifier. Then, we details the confidence measure and rejection capacity we introduced to develop supervision strategies using user interactions.

### A. System Architecture

We focus here on Fuzzy Inference Systems (FIS) [10], with first order conclusion structure – so-called Takagi-Sugeno FIS [11]. FIS have demonstrated their good performances for incremental classification of changing data flows [12]. Moreover, they can easily be trained on-line – in real time – and have a good behavior with new classes. In this section, we present the architecture of the evolving FIS *Evolve* [6] that we use to recognize our gesture commands.

Fuzzy Inference Systems consist of a set of fuzzy inference rules like the following rule example.

$$\textbf{Rule}^{(i)} : \textbf{IF } \mathbf{x} \text{ is close to } C^{(i)} \tag{1}$$

$$\textbf{THEN } \hat{\mathbf{y}}^{(i)} = (\hat{\mathbf{y}}_1^{(i)} ; \ldots ; \hat{\mathbf{y}}_c^{(i)})^\top \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector, $C^{(i)}$ the fuzzy prototype associated to the *i*-th rule and $\hat{\mathbf{y}}^{(i)\top} \in \mathbb{R}^c$ the output vector. Rule premises are the fuzzy membership to rule prototypes, which are clusters in the input space. Rule conclusions are fuzzy membership to all classes, that are combined to produce the system output.

*1) Premise Structure:* Our model uses rotated hyper-elliptical prototypes that are each defined by a center $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^n$ and a co-variance matrix $\Sigma^{(i)} \in \mathbb{R}^{n \times n}$ (where $n$ is the number of features).

The activation degree $\alpha^{(i)}(\mathbf{x})$ of each fuzzy prototype is computed using the multivariate normal distribution.

*2) Conclusion Structure:* In a first order FIS, rule conclusions are linear functions of the input:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x}) ; \ldots ; l_c^{(i)}(\mathbf{x})) \tag{3}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \cdots + \theta_{n,k}^{(i)} \cdot x_n \tag{4}$$

The *i*-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{5}$$

with $\Theta^{(i)} \in \mathbb{R}^{n \times c}$ the matrix of the linear functions coefficients of the *i*-th rule:

$$\Theta^{(i)} = (\boldsymbol{\theta}_1^{(i)} ; \ldots ; \boldsymbol{\theta}_c^{(i)}) \tag{6}$$

*3) Inference Process:* The inference process consists of three steps:

1) Activation degree is computed for every rule and then normalized as follows:

$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{7}$$

where $r$ is the number of rules.

2) Rules outputs are computed using Equation 5 and system output is obtained by sum-product inference:

$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{8}$$

3) Predicted class is the one corresponding to the highest output:

$$\text{class}(\mathbf{x}) = \arg \max_{k=1}^{c} (\hat{y}_k) \tag{9}$$

Figure 2 represents a FIS with first order conclusion structure as a radial basis function (RBF) neural network.

### B. Incremental Learning Process

Let $\mathbf{x}_i$ ($i = 1..t$) be the $i$-th data sample, $M_i$ the model at time $i$, and $f$ the learning algorithm. The incremental learning process can be defined as follows:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{10}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{11}$$

In our recognizer *Evolve* [6], both rule premises and conclusions are incrementally adapted:

1) Rule prototypes are statistically updated to model the run-time data:

$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1)\cdot \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{12}$$

$$\Sigma_t^{(i)} = \frac{(t-1)\cdot \Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{13}$$

2) Rule conclusions parameters are optimized on the data flow, using the Recursive Least Squares (RLS) algorithm:

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)} \mathrm{C}_t^{(i)} \mathbf{x}_t (\mathbf{y}_t^\top - \mathbf{x}_t^T \Theta_{t-1}^{(i)}) \tag{14}$$

$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)} \mathbf{x}_t \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)} \mathbf{x}_t} \tag{15}$$

New rules, with their associated prototypes and conclusions, are created by the incremental clustering method *eClustering* [13] when needed.

### C. Confidence Measure and Rejection Threshold

We use confusion reject principles to evaluate the system confidence of recognized labels. Usually, confusion reject is based on system output (membership to all classes). However, we try to detect confusion, to evaluate our model quality, at a very early stage of the on-line learning process. As a result, inference rules conclusions are still rough and unstable, and not very representative of the system confidence. Instead, we choose to use rules premises which are much more stable at this early stage of the on-line training. Even though every prototype participates in the recognition of every classes, each prototype has been created by and is mainly associated with a single class. We use that fact to detect confusion when some gesture activates different prototypes at similar levels.

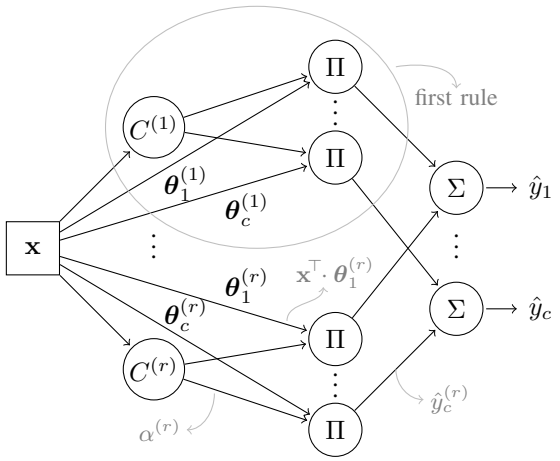


Fig. 2. First order FIS as a radial basis function (RBF) neural network

We use the Mahalanobis distance to compute the distance of a data sample $\mathbf{x}$ to the prototypes $C^{(i)}$ (defined by their center $\boldsymbol{\mu}^{(i)}$ and co-variance matrix $\Sigma^{(i)}$).

$$distance(C^{(i)}, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top \tag{16}$$

From this distance, we compute similarity measures that are smoother than prototype activation.

$$similarity(C^{(i)}, \mathbf{x}) = \frac{1}{1 + distance(C^{(i)}, \mathbf{x})} \tag{17}$$

With these similarity measures, we compute system confidence as:

$$confidence = \frac{s_{first} - s_{second}}{s_{first}} \tag{18}$$

Where $s_{first}$ and $s_{second}$ are the first and the second highest similarity values. A data sample is then signaled as confusing when its confidence is below a certain threshold.

The optimization of the rejection threshold is a multi-objective problem: we want to maximize both classifier performance and accuracy.

$$Performance = N_{Correct}/N_{Total} \tag{19}$$

$$Accuracy = N_{Correct}/(N_{Correct} + N_{Errors}) \tag{20}$$

Where $N_{Correct}$ is the number of correctly classified gestures, $N_{Errors}$ is the number of incorrectly classified gestures, and $N_{Total}$ is the total number of gestures. As the threshold increases, the number of rejected gestures raises and the number of classification errors reduces.A high threshold will yield many rejections, which will increase system accuracy, whereas a low threshold will yield only a few rejections, which will increase system performance. There is a trade-off between the classifier performance and accuracy.

To solve this trade-off, we must define the cost of an error of classification, and the cost of a rejection. One the one hand, a rejection will make the system ask the user to validate or correct the recognized label. On the other hand, an error of classification will force the user to cancel/undo his command and do it again. Our goal is to reject data that don't fit well into the classifier model, to reduce classification errors but also to improve its model. However, we don't want to reject too many data and solicit the user to often.

## III. Supervision Strategies

In the context of gesture commands, users initialize the system with a few gestures per class (three in our experimentation). To improve gesture command recognition, the classifier learns incrementally during its use.

At the same time that the classifier is learning, so is the user: he has to memorize which gesture is associated with

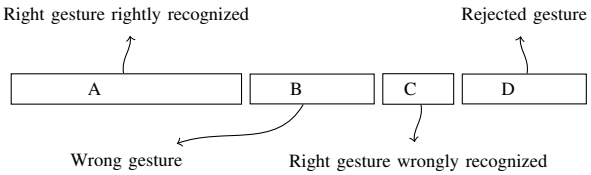


Fig. 3. Data partitioning as a result of the user and system cross-learning

which command [4]. In this cross-learning situation, different cases can happen:

| | |
|---|---|
| Case A | The user draws the right gesture which is rightly recognized: the intended command is executed; |
| Case B | The user makes a mistake and draws a wrong gesture; |
| Case C | The classifier makes a mistake and recognizes a wrong label/command; |
| Case D | The classifier rejects the gesture and asks the user to confirm or correct its recognition. |

When either the user or the system makes a mistake, the command which is executed is not the one that was intended. The user has to cancel/undo that command and try again to do the one he wants. Data can be divided into four categories like shown in Figure 3.

The on-line learning algorithm used to train the classifier during its use is a supervised algorithm. It is hence necessary to label run-time data. Two strategies are used: an implicit strategy, without interacting with the user, and an explicit strategy, that solicits the user to obtain data true label.

### *A. Supervision strategy based on implicit labeling, without user interaction*

The advantage of implicit strategies is that they don't disturb users during their use of gesture commands. The implicit supervision strategy consists of labeling run-time data with labels recognized by the classifier, without interacting with the user. However, this labeling will contain mistakes each time gestures are wrongly recognized. To avoid deteriorating the classifier model by learning on mislabeled data, we take advantage of the user next action, to learn only when he implicitly validates the recognized label.

In practice, when the user draws a gesture, it is recognized by the classifier and the corresponding command is executed. Two cases are then possible.

- The user cancels or undo this command, either because it doesn't correspond to the gesture he has drawn (classification error, category C of Figure 3), either because he has drawn a wrong gesture (memorization error, category B), or just because he changed his mind.
- The user continues his actions, which is likely to indicate that the executed command suit his needs, he implicitly validates the recognition (category A).

The implicit strategy is to use the recognized label, but to learn only when it is implicitly validated by the user (by doing another command that cancel/undo). The classifier will learn from the data samples it has correctly recognized (category A), but it will not learn from his mistakes (category C), nor from the user mistakes (category B), rather than risking to learn with a wrong label. This strategy allows to be sure not to deteriorate the classifier model, but reduces the number of data that can be used for the on-line training. Furthermore, the classifier only learns from data that are correctly recognized, learning from them is interesting but not as much as learning from data that are incorrectly classified.

### *B. Supervision strategy based on explicit labeling by user interactions*

Learning from incorrectly recognized data requires to interact with the user to obtain the true label of the gesture he has drawn. It seems obvious that soliciting the user after each command would be very tedious for the user. We must carefully select the data samples we ask him to label. To do so, we use the classifier confidence measure to select the data samples that aren't well described by the classifier model, and from which it will be very beneficial to learn. By doing so, the classifier can learn from the gestures that are complex to recognize (category D of Figure 3), and for which it would have probably made a mistake.

Overall, data from categories B and C of Figure 3 aren't used because their label hasn't been validated, neither explicitly nor implicitly. Only data from categories A (implicitly validated) and D (explicitly validated) are used for the on-line training. This choice takes out a few data that aren't used for training the classifier, but allow to be sure not to deteriorate its model by learning on potentially mislabeled data. As a consequence, the rejection strategy has a great influence on the classifier training process. The more data are rejected, the more data are available to train the classifier. Besides, learning from rejected data is very beneficial for the classifier model.

*1) Using a Constant Rejection Threshold:* To optimize system training, and hence performances, we can tune the rejection threshold to increase data labeling, but we need to keep in mind its impact on user interactions. It is necessary to find a good compromise between the number of recognition errors, and the number of rejection/user interactions. The most simple explicit supervision strategy is to choose a constant rejection threshold to optimize the error/reject trade-off in a classical way. The data samples with the highest probability of being misrecognized are rejected, and rejected data are homogeneously distributed over time. However, we have here an evolving system that will improve with time, as a consequence, the number of rejected data will decrease with time.

*2) Using a Varying Rejection Threshold:* The error/reject trade-off is quite complex in this on-line learning situation. Rejecting some data that would have been misrecognized not only represent a mistake avoided, but also an additional training data to improve the system. The intuition that we have is that for the same number of rejections, we will obtain a better system if we concentrate those rejections at the beginning of the utilization/training (to accelerate the learning process). We thus propose a second strategy based on explicit labeling by the user but using a varying threshold. This strategy

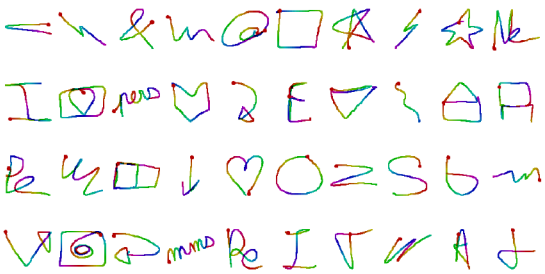

Fig. 4. Gesture samples of ILGDB (group 1: free gestures)

of a varying threshold enable to have a high rejection rate at the beginning of system use, to quickly reduce its error rate, and then to reduce the rejection rate to minimize the number of user interactions, and to end up with an optimal operating point for the future use of the system.

## IV. Experimentation

Our objective is to improve our classifier recognition performances as much as possible, but without soliciting the user too often. We compared experimentally the supervision strategies with a constant rejection threshold (cf. section III-B1) and with a varying threshold (cf. section III-B2) to accelerate the learning at the beginning of system use.

### A. Evaluation Protocol

We evaluated the different supervision strategies on the ILG Data Base[2] [14] using the supplied HBF49 [15] feature set. ILGDB contains 6629 mono-stroke gestures, belonging to 21 classes, that have been drawn by 38 writers in an immersive environment. This database is very interesting for three reasons. First, gestures are chronologically ordered (in their drawing order) which enable to see the evolution of users writing styles with time. Second, class frequencies vary, from 5 to 17 samples per class (per writer). Third, for the majority of the database, gesture classes where freely chosen by the writers themselves. These three reasons make this database very realistic and representative of the real use of a handwritten gesture on-line classifier. Furthermore, the low number of samples per writer (less than 180), and per class (less than 20), makes this database a challenging benchmark for evolving classifiers. Some gesture samples invented by ILGDB writers are presented in Figure 4.

Drawn symbols are distributed into five phases for each writer. Phase 0 contains three symbols per class, and phases 1 to 5 ($\sim$ 120 symbols) correspond to system use varying class frequencies. In order to simulate a longer use of gesture commands, we used 20 random triplets of users that use the same gestures (group 3). We initialize our system on phase 0 of the first writer and use phases 1 to 3 of the three writers ($\sim$ 270 symbols) to simulate the on-line training of our recognizer. We tested our system performances on phase 4 of the three writers (63 symbols) between each of the nine utilization/training phases (see Figure 5).

[2]Freely available at http://www.irisa.fr/intuidoc/ILGDB.html

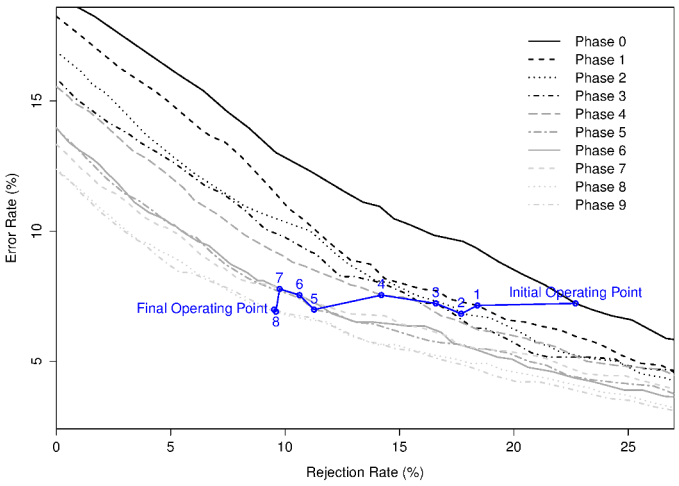


Fig. 5. Evolution with time (constant threshold of 0.2)

### B. Comparison of Implicit and Explicit Supervision Strategies

Table II present the final error rates obtained on ILGDB with the implicit (cf. section III-A) and explicit (cf. section III-B) supervision strategies. As expected, learning without user supervision, neither implicit nor explicit, but using the recognized labels only (self supervision), deteriorates the classifier performances. It is better not to learn at all than to learn on potentially mislabeled data that damage the classifier model. The implicit supervision strategy allows to improve recognition performances, but not as much as the explicit strategy. It is essential for the classifier to be able to learn from its mistakes. Finally, combining the implicit strategy with the explicit one allow to improve efficiently the classifier model and its performances.

### C. Evolution of the Error/Reject Curve with Time

Figure 5 present the evolution of the error/reject curve with time for a constant rejection threshold of 0.2. Each curve represents the error/reject trade-off reached by the classifier after the corresponding utilization/learning phase. The error/reject curves improve with time, both the error rate and the rejection rate decrease, as the classifier learns and improves during its use. We can notice that this diminution is fastest at the beginning of the utilization/training, and that the error/reject curve ends up stabilizing.

*Final erreur/rejet curves:* Figure 6 present the final error/reject curves obtained with a constant threshold of 0.2 and 0.3, and with a varying threshold from 0.3 to 0.15. The variations of the threshold are quite rough here, we only make it vary by steps for demonstration purposes. We use a threshold

TABLE II. Comparison of implicit and explicit supervision strategies on ILGDB

| Supervision Strategy | Final Error Rate (%) |
|---|---|
| No learning | 11.53 |
| Self supervision | 13.28 |
| Implicit supervision | 10.15 |
| Explicit supervision | 7.27 |
| Combination of implicit and explicit supervision | 6.52 |

TABLE I. AVERAGE NUMBER OF USER INTERACTION PER WRITER (ON THE 21 GESTURES OF PHASE 4)

| Test after Phase | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Constant Threshold of 0.2 | 4.77 | 3.87 | 3.71 | 3.48 | 2.98 | 2.37 | 2.23 | 2.05 | 2.02 | 2.00 | 88.5 |
| Constant Threshold of 0.3 | 6.87 | 5.83 | 5.35 | 5.25 | 4.73 | 4.07 | 3.98 | 3.95 | 3.72 | 3.65 | 142 |
| Varying Threshold from 0.3 to 0.15 | 6.87 | 5.83 | 5.35 | 5.25 | 3.90 | 3.40 | 3.17 | 1.67 | 1.55 | 1.57 | 115 |

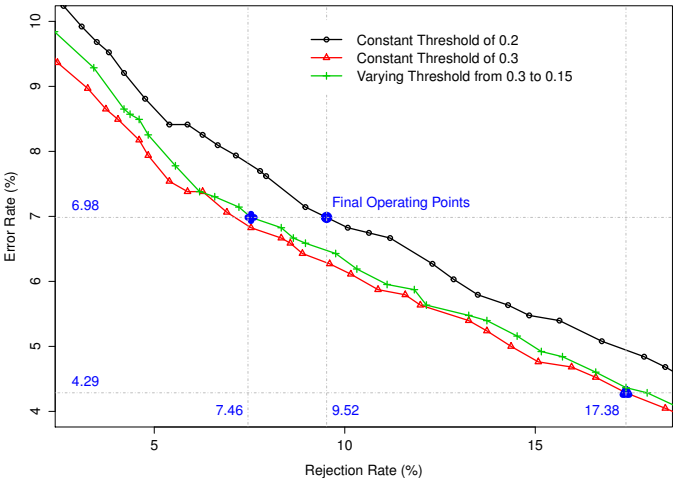


Fig. 6. Final Curves for the Different Strategies

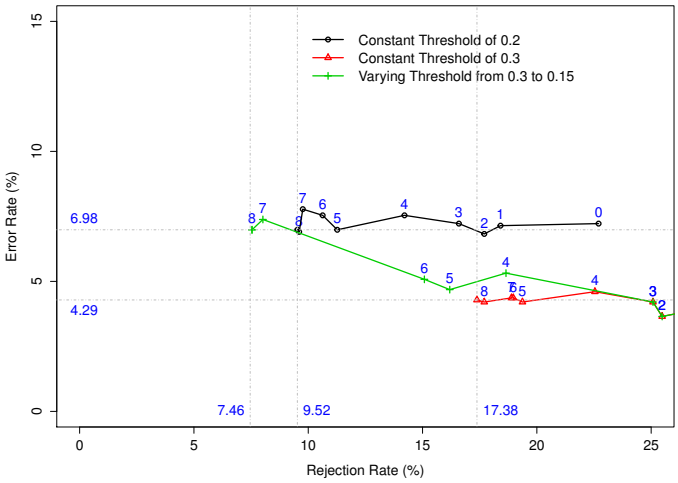


Fig. 7. Operating Points Evolution for the Different Strategies

of 0.3 for the three first phases, a threshold of 0.25 for the three next, and finally a threshold of 0.15 for the last phases.

A high constant rejection threshold (like 0.3) allows to reach better performances than a lower threshold (like 0.2), however the operating point obtained with this higher threshold is far from optimal as it solicits a lot the user (17.38%). Using a varying threshold enable to obtain a final error/reject curve as good as a high rejection threshold, and at the same time guarantee a final operating point that is optimal from the user interaction point of view. Rejecting a lot of data at the beginning of system use allow to accelerate the classifier training, and reducing it when the recognizer performances have converged allow to have an optimal operating point for the rest of the command gesture system utilization.

### D. Operating Point Evolution

Figure 7 present the evolution of the operating points obtained with a constant threshold of 0.2 and 0.3, and with a varying threshold from 0.3 to 0.15. Operating points move from right to left, the rejection rate decreases as the system improves and confidence scores increase. Having a high rejection rate at the beginning of the utilization/training allow to accelerate system learning process, and to obtain, when we reduce the rejection rate, an operating point with a lower rejection rate (7.46% instead of 9.52%, 22% of relative diminution) for a similar error rate (6.98%). Using a varying rejection threshold optimizes the cooperation between the user and the system for the on-line training of an evolving recognizer. It allows to reach a better operating point than those obtained with static thresholds, and without penalizing the user system cooperation.

*User Interaction Evolution:* Table I present the average number of user interaction after each utilization/training phase. The average number of solicitations on the whole experiment goes from 142 for the constant threshold of 0.3, to 115 for the varying threshold (23% of relative diminution), and with a similar error rate. Increasing by 31% the number of user interactions at the beginning (with respect to a constant threshold of 0.2) enable to decrease it by 21% at the end of the experiment, and thus for the future use of the command gesture system.

## V. CONCLUSION

Training a classifier for the recognition of gesture commands is an on-line learning situation that requires a supervision strategy to label run-time data. Most of the correctly recognized data can be labeled implicitly with users next action, but it is essential to be able to learn from misrecognized data. To do so, it is necessary to interact with the user to be able to label complex data and improve our classifier model efficiently. On the other hand, constantly soliciting the user is tedious, and considerably reduces the easiness of use of gesture commands. A compromise must be chosen between the number of user interactions and the number of recognition errors.

We have studied the impact of different rejection strategies to supervise the on-line training of an evolving recognizer for gesture commands, and how to optimize this cooperation between the user and the recognition system. As it is fundamental to be able to learn from misrecognized data samples with their correct labels to improve the classifier performance, we try to obtain the best performance from the classifier with as few user interactions as possible. In particular, we use an inner confidence measure to solicit the user when some data samples don't fit with the classifier model, and that it will be very

gainful to learn from it, but without interacting to often. In this way, using a varying rejection threshold allows to solicit more often the user aThis new supervision strategy, with a varying rejection threshold, improves our recognition system performances on the experimentation we conducted from an existing database. It would now be interesting to study its impact in real use and in particular its consequences on users behavior. Moreover, conducting some experimentations on a longer period of time, with the same user, should increase the advantage of the varying threshold strategy presented in this article.t the beginning of system use, which accelerates the learning process and thus improves classifier performances. This faster improvement of the classifier performance enables to then reduce the rejection rate, because the classifier makes less mistakes, and reduce user interactions.

We have compared experimentally two supervision strategies of the learning process using rejection to explicitly label complex data, some labeling more data and some soliciting less often the user. The first strategy is to use a constant rejection threshold, whereas the second one is to use a varying threshold that start with a high value and then reduces to a lower one. This second strategy allows to accelerate the learning process of the classifier at the beginning of its use by labeling more data. The error/reject curve and the operating point obtained in this way are more interesting: the rejection rate is $22\%$ lower for the same error rate. Furthermore, increasing the number of user interactions at the beginning of system utilization is a mean to keep the user in the loop. It maintains his vigilance and encourages him to spontaneously correct the system as frequently as necessary, which is essential to improve the classifier performances as much as possible.

This new supervision strategy, with a varying rejection threshold, improves our recognition system performances on the experimentation we conducted from an existing database. It would now be interesting to study its impact in real use and in particular its consequences on users behavior. Moreover, conducting some experimentation on a longer period of time, with the same user, should increase the advantage of the varying threshold strategy presented in this article.

## REFERENCES


[1] J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1083–1092.

[2] J. Yang, J. Xu, M. Li, D. Zhang, and C. Wang, "A real-time command system based on hand gesture recognition," in *2011 Seventh International Conference on Natural Computation (ICNC)*, vol. 3, 2011, pp. 1588–1592.

[3] P. Y. Li, N. Renau-Ferrer, E. Anquetil, and E. Jamet, "Semi-customizable gestural commands approach and its evaluation," in *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 473–478.

[4] P. Li, M. Bouillon, E. Anquetil, and G. Richard, "User and system cross-learning of gesture commands on pen-based devices," in *Proceeding of the 14th International Conference on Human-Computer Interaction - INTERACT 2013*, vol. 2, 2013, pp. 337–355.

[5] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ser. UIST '07. New York, NY, USA: ACM, 2007, pp. 159–168.

[6] A. Almaksour and E. Anquetil, "Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers," *Evolving Systems*, vol. 2, no. 1, pp. 25–33, 2011.

[7] ——, "ILClass : Error-driven antecedent learning for evolving takagi-sugeno classification systems," *Applied Soft Computing*, 2013.

[8] D. B. Kaber and M. R. Endsley, "Out-of-the-loop performance problems and the use of intermediate levels of automation for improved control system functioning and safety," *Process Safety Progress*, vol. 16, no. 3, pp. 126–131, 1997.

[9] D. A. Norman, "The 'problem' with automation: inappropriate feedback and interaction, not 'over-automation'," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 327, no. 1241, pp. 585–593, 1990.

[10] E. Lughofer, *Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications*. VDM Verlag Dr. MÃijller, 2008.

[11] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man, and Cybernetics, IEEE Transactions on*, vol. 15, no. 1, pp. 116–132, 1985.

[12] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008.

[13] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 484–498, 2004.

[14] N. Renau-Ferrer, P. Li, A. Delaye, and E. Anquetil, "The ILGDB database of realistic pen-based gestural commands," in *Proceeding of the 21st International Conference on Pattern Recognition*, 2012, pp. 3741–3744.

[15] A. Delaye and E. Anquetil, "HBF49 feature set: A first unified baseline for online symbol recognition," *Pattern Recognition*, vol. 46, no. 1, pp. 117–130, 2013.